Design and Implementation of an M/G/1 Queue with Working Vacations using Genetic Algorithm

Daniel Naicker, *Student Member, IEEE,* Mduduzi C. Hlophe, *Member, IEEE,* and Bodhaswar T. Maharaj, *Senior Member, IEEE*

***Abstract*—In real-world contexts, most systems are not one hundred percent reliable. Breakdowns in the service area occur due to external shocks, which has given rise to the study of stochastic models with specific research into server breakdowns in queuing theory [1]. Queuing models with backup servers to the primary server have found many applications in computer systems and wireless communications. When the primary server breaks down, the backup server takes over and acts as the primary server working at a reduced rate during a working vacation. However, the sensitivity of the parameters on various performance indices is difficult to explore since the parameters of optimality are static. In order to address this issue, this article studies the performance model for an $M/(M_1, M_2)/1$ queue with working vacation and vacation interruptions, which is implemented and improved using a genetic algorithm (GA). Each of the parameters of the queuing model and its effects on the queue length were investigated and it was found that the GA produced improved results and allowed for more flexibility in the design of the queuing system. The proposed GA model offered an average of 10.79% lower expected queue length. These investigations conducted through simulations indicate that the proposed GA can be used with the queuing model presented to provide an easier way to design $M/(M_1, M_2)/1$ queuing systems with working vacations and vacation interruptions with lower queue lengths.**

***Index Terms*—Expected length, Genetic algorithm, $M/G/1$, $M/(M_1, M_2)/1$, Performance analysis, Queuing theory, Server, Working vacations.**

Open License: CC-BY

I. Introduction

QUEUING theory is a mathematical framework that is used to analyze the performance of waiting lines. Since excessive waiting would incur costs related to delays, the primary objective is to reduce the adverse effects of waiting to tolerable levels that meet the design specifications. It is not an optimization technique on its own, but rather used alongside optimization techniques to achieve optimal system performance [2]. Queuing theory techniques have been applied in various fields, such as telecommunications, manufacturing, transportation, as well as in healthcare. For instance, its applications in telecommunications are mainly in the optimization of traffic flow with the aim of reducing congestion in routers [3]. It is specifically used to improve the performance of computer networks by regulating the number of requests or packets transmitted through network channels [2]. Also, in healthcare, queuing theory has been used to enhance patient flow in emergency departments and hospitals [4]. In manufacturing, it is used to optimize production lines, inventory management, and quality control [5].

The formation of queues occurs when the average arrival rate is greater than the average service rate. The design of systems, such as communication switches and routers, base stations, sensor networks, servers, and road traffic intersections, among others, uses queuing theory. The key goal of queuing theory is to determine and control system performance under different scenarios, including queue length, waiting time, busy period, and loss probability. These parameters are crucial in regulating arrival or service rates, adjusting queue discipline, buffer size, and the number of service channels, among other factors [6]. In the context of queuing theory, vacations are periods when the server is unavailable, which can occur mainly due to preventative maintenance. In this case, the combination of queuing theory and vacations is essential in the design and analysis of computer systems, communication networks, and production management [7]. The most prevalent queuing model in queuing theory is the $M/M/1$ model, which is a basic model used in queuing theory to describe a single server model consisting of Poisson arrival rates and Poisson service rates. The first 'M' represents the arrival process that is Markovian which means that the interarrival times are exponentially distributed, the second 'M' represents the service process that is Markovian which means that the times are exponentially distributed, and lastly, the '1' represents the amount of servers in the system which is a single server in this case. In other words, the arrival times of customers are random and can be modeled using a Markovian process. Working vacations is an approach that makes use of vacation periods which takes advantage of the lower server utilization rate to save resources while still allowing the server to operate [8]. In this way, the server is allowed to take breaks during periods of low customer arrival rates, which has great relevance in various applications such as service-oriented systems and maintenance scheduling.

A. Previous Research Works

A technique that combines the $M/M/1$ model, working vacations, and two types of server breakdowns was investigated in an earlier study [9], which provided insight into the behavior and performance of queuing systems that contain

Daniel Naicker, Mduduzi C. Hlophe, and Bodhaswar T. Maharaj are with the Department of Electrical, Electronic and Computer Engineering, University of Pretoria, Pretoria, South Africa, e-mail: u18006192@tuks.co.za, mduduzi.hlophe@tuks.co.za, and sunil.maharaj@up.ac.za.

This work was supported by the Sentech Chair in Broadband Wireless and Multimedia Communication (BWMC), University of Pretoria.

service interruptions and breakdowns. The study proposed a framework that used servers on vacation by running them at a lower service rate where both minor and major server breakdowns were considered. The research conducted on this model provided derivations for many of the important design parameters used in current queuing techniques with working vacations such as steady-state probabilities, mean system size, and mean queue length. The authors of the study explored the impact of the parameters within the derivations on the system which offer valuable guidelines for designing robust and efficient queuing systems with working vacations. These design guidelines are useful for environments that are prone to server interruptions. In the contributions in [10], [11], the authors analyzed the mean waiting time and mean queue length as the main performance metrics to understand the impact of working vacations and vacation interruptions on system performance. The authors in [10] derived expressions for the steady state characteristics of the system and analyzed various parameters for performance, such as the mean waiting time and system availability. Then, in [11] they contributed to the understanding and optimization of queuing-inventory systems using the same performance metrics. A study focusing on the equilibratory and optimality of $M/M/1$ queues was conducted in [12], where the expressions for the given system in terms of the expected reward rate and optimal policies were derived. The studies in [11] and [12] were extended in [13] to analyze the steady state of not only the $M/M/1$ queue, but also the $D/M/1$ and $M/G/1$ queuing models. Here, the performance metrics derived from earlier $M/M/1$ systems were used for $D/M/1$ and $M/G/1$ queues.

In [14], a functional analysis method for the $M/G/1$ queuing model with single working vacations was then derived with a focus on the stability of the system. The performance measures of the system were analyzed using functional analysis techniques and it was observed that the time-dependent solution of the queuing model strongly converges to its steady-state solution. The authors in [15] analyzed the behavior of a multi-server Markovian working vacation queue with breakdowns, where they derived expressions for the performance measures such as the mean waiting time and system availability. Contrary to the traditional $M/M/1$ queue, a multi-server system involves multiple servers available to serve customers simultaneously. An important aspect of their study was that service rates depend on the state of the server, allowing for a more realistic representation of varying service capacities. The derived expressions for multi-server Markovian working vacation queues are significant as queuing models with multiple servers could be optimized to increase service capacity, throughput, and scalability.

Although the $M/G/1$ and $M/(M_1, M_2)/1$ queues differ in terms of the number of servers and service time distributions, they are related in that the $M/G/1$ queue can be regarded as a special case of the $M/(M_1, M_2)/1$ queue. This is because when the two types of servers in the $M/(M_1, M_2)/1$ queue have the same service rate, the $M/(M_1, M_2)/1$ queue simplifies to the $M/G/1$ queue [16]. Hence, the $M/G/1$ queue can be viewed as a simplified version of the $M/(M_1, M_2)/1$ queue, which is more versatile and general in modeling queuing systems [17].

In this study, the focus is on a specific queuing model, namely the M/(M1,M2)/1 queuing model with working vacations and vacation interrupts, which has been proposed in [8]. An M/(M1,M2)/1 queue is a special case of an M/G/1 queue, where the arrival rate is exponentially distributed, and the service times for the primary and backup servers are exponentially distributed with rates μ_1 and μ_2, respectively, and the probability of the backup server being used is ρ. This model enables analysis of the performance of the queuing system, including the expected queue length, state probability, and waiting time. The model is based on the M/G/1 queuing model, where the arrival rate is exponentially distributed and the service times for the primary and backup servers are exponentially distributed with rates μ_1 and μ_2. This model provides a comprehensive framework for understanding and optimizing queuing systems that need to operate under realistic interruption conditions for both minor and major server breakdowns.

B. Research Motivation and Summary of Contributions

The queue length and service status for an arbitrary epoch under steady-state conditions were determined using a supplementary variable and matrix-analytic method as in [8]. However, the M/G/1 queuing model with working vacations and vacation interrupts provides a useful tool for analyzing and designing queuing systems with backup servers, and it offers a set of computable performance measures. The limitations of this model for designing a queuing system are that the exact design parameters are required and that it does not offer a solution for an optimal design of the system. The motivation of this study is to address this limitation by developing an algorithm that makes use of the proposed queuing model in [8] to aid in the optimal design of an M/G/1 queue with working vacations and vacation interruptions. The contribution of this paper can be summarized below,

- **Design and Implementation:** An investigation on the design and implementation of a genetic algorithm (GA) for the $M/(M1, M2)/1$ queuing model with working vacation and vacation interruptions was carried out. The motivating assumption for taking this approach was that the GA M/(M1,M2)/1 queuing model would provide flexibility in the design and also lower the expected queue length by allowing a user to enter a range of values for the queuing system. This approach is contrary to popular approaches which use predefined static values in order to obtain the optimum result. The results of the proposed GA $M/(M1, M2)/1$ show that the proposed approach lowers the expected queue lengths compared to the base queuing model by 10.79%.
- **Investigating Input Parameters:** An investigation of the effects of each design parameter was carried out in order to guide the design decisions when using the GA queuing model. Each of the parameters of the queuing model and its effects on the queue length was investigated and this insight can be used to determine the appropriate input ranges for optimal performance. The objective is to

identify the combination that yields the minimum queue length, indicating an optimal configuration for the system. This proposed model provides a methodical approach to parameter tuning which optimizes each of the design parameters within the GA, leading to improved queue efficiency by 10.79% and overall system performance.

This article is organized as follows: Section **II** discusses the proposed system model, while the queuing model and its notational implementation are discussed in Section **III**. Section **IV** reviews the proposed GA model with respect to queuing theory, design, and algorithmic implementation. Section **V** evaluates the proposed model by using performance simulation. Finally, Section **VI** gives the concluding remarks together with the Advantages and suggestions for future research.

II. Proposed System Model

A summary of the definitions and mathematical expressions for the queuing model is shown in the table below:

TABLE I: Mathematical Symbols for Base Queuing Model

Symbol	Description
$S_b(x)$	PDF of the normal service time during the busy period
μ_b	Mean service rate during the busy period
$S_b^*(r)$	Laplace-Stieltjes Transform of $S_b(x)$
$S_v(x)$	PDF of the service time during the vacation period
μ_v	Mean service rate during the vacation period
$S_v^*(r)$	Laplace-Stieltjes Transform of $S_v(x)$
$E(L)$	Mean expected queue length

The operation of the proposed GA is illustrated in Fig. 1:

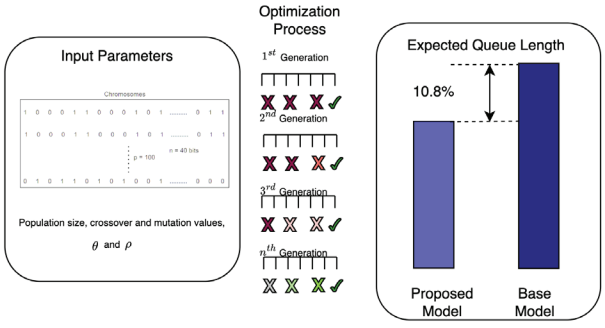


Fig. 1: Illustration of the optimization process of the genetic algorithm.

As shown in Fig. 1, the input parameters used in the optimization process are: (i) population size, (ii) the number of mutations, (iii) the number of crossover pairs, (iv) traffic intensity (ρ) values, and (v) vacation rate (θ), meaning how many time does the system go into vacation. The crucial factor lies in the range of the ρ and θ values, which defines the exploration space for the parameters. By carefully specifying their range, the model can systematically search for optimal parameter combinations through an iterative process. The final step of the model is to evaluate each parameter set and calculate the resulting queue length. The system model can then be applied to a GA which would optimize the ρ and θ values to minimize the expected queue length.

III. Preliminaries and Model Definition

In this section, the design and implementation of the queuing model and the expected queue length are discussed with the assumption of a first-in-first-out (FIFO) service discipline. The FIFO service is important in understanding how the queuing model operates due to it directly influencing the order in which the servers are used. Traditionally, in a FIFO system customers are served in the precise order that they arrive with all customers served equally, which is a common and straightforward service rule within many real-world applications. The equality within the system allows for the focus to be on the effects of other parameters like arrival rates, service rates, and the vacation policies of the server. The relationship between the queuing model and the service discipline is vital as it ensures that the analysis of the queue's performance is both consistent and predictable. The model is parametrized by:

- the service time distribution for the busy server, $S_b(x)$, with finite mean μ_1,
- the vacation distribution, $S_v(x)$, with finite mean μ_0.

A. Expected Queue Length of Base Queuing Model

Assuming that the service discipline for the base queuing model FIFO, and the probability distribution function (PDF) can be split into service times during the busy period and the service times during the vacation period. The PDF for the normal service time for the busy server can be expressed as follows [18]:

$$S_b(x) = 1 - exp\left\{-\int_0^x \mu_1(t)dt\right\}, \tag{1}$$

with a mean service rate of $1/\mu_b$, where

$$S_b^*(r) = \int_0^\infty e^{-rx} dS_b(x), \quad \text{and} \quad b^{(k)} = \int_0^\infty x^k dS_b(x).$$

The PDF for the vacation period is given as follows:

$$S_v(x) = 1 - exp\left\{-\int_0^x \mu_0(t)dt\right\}, \tag{2}$$

with a mean service rate of $1/\mu_v$, where

$$S_v^*(r) = \int_0^\infty e^{-rx} dS_v(x). \tag{3}$$

In order to get the steady-state conditions, assuming that λ is the arrival rate, the inequality $\rho = \frac{\lambda}{\mu_b} < 1$ is a necessary and sufficient condition for the system to be stable. Using this information and solving for the model by solving the resulting partial differential equations, the partial periodic Green's function (PGFs) and marginal PGFs. Given that θ is the rate of vacation time, the mean expected queue length (L) is derived as follows:

$$E(L) = \frac{\lambda+\theta}{\theta} + \frac{\lambda^2 b^{(2)}}{2(1-\rho)} - \frac{(1-\rho)(\lambda+\theta)+(\lambda+\rho\theta)\beta}{\theta(1-\rho)+(\lambda+\rho\theta)[1-S_v^*(\theta)]} - \frac{\theta\lambda^2 S_v^*(\theta) b^{(2)}}{2\{\theta(1-\rho)+(\lambda+\rho\theta)[1-S_v^*(\theta)]\}}, \tag{4}$$

where ρ is the service intensity given in [8], and $\beta = \int_0^\infty \lambda x e^{-\lambda x} dS_v(x)$.

B. Implementation of Base Queuing Model

The implementation of the model for the M/G/1 queue with working vacations and vacation interruption assumes an exponential distribution (M_1 and M_2) for the service times during the busy period and the service times during the working vacation which results in an $M/(M_1, M_2)/1$ queuing model. The normal service times during the busy period has a rate of $\mu_b = 2$, while the mean for the service times during the vacation period is $\frac{1}{\mu_v}$. The PDF and cumulative density function (CDF) for the exponential distribution can be expressed as follows:

$$PDF = \lambda e^{-\lambda t}, \quad \text{and} \quad CDF = 1 - e^{-\lambda t}. \tag{5}$$

The mean of the exponential PDF is $\frac{1}{\lambda}$ and the mean for the service times during the vacation period is $\frac{1}{\mu_v}$ which results in $\mu_v = \lambda$. The mean for the service times during the vacation period is $\frac{1}{\mu_b}$ which results in $\mu_b = \lambda$. Using the equations in (5), μ_0 and μ_1 can be computed using

$$u_0 = \mu_b e^{-\mu_b t} \quad \text{and} \quad u_1 = \mu_v e^{-\mu_v t}. \tag{6}$$

Substituting u_1 and u_0 into the equations for the PDF of the busy server $S_b(x)$ in (1) and the PDF of the vacation server $S_v(x)$ in (2) respectively, which results to

$$S_v(x) = 1 - e^{e^{-\mu_v x}-1} \quad \text{and} \quad S_b(x) = 1 - e^{e^{-\mu_b x}-1}. \tag{7}$$

In order to obtain the values for β, $b^{(2)}$ and $S_v^*(\theta)$, let $g(x)$ be continuously differentiable in x so that the following identity can be applied:

$$\int f(x)d(g(x)) = \int f(x)g'(x)dx. \tag{8}$$

Using this identity and noting that $S_v(x)$ and $S_b(x)$ are continuously differentiable in x, the equations for β, $b^{(2)}$ and $S_v^*(\theta)$ can be obtained as follows:

$$\begin{aligned} \beta &= \int_0^\infty \lambda x e^{-\lambda x} \times S_v(x)' dx \\ b^{(2)} &= \int_0^\infty x^2 \times S_b(x)' dx, \\ S_v^*(\theta) &= \int_0^\infty e^{-\theta x} \times S_v(x)' dx. \end{aligned} \tag{9}$$

After substitution (1) and (2) into the relevant equations in (9), and further differentiation, the final equations for β, $b^{(2)}$ and $S_v^*(\theta)$ can be expressed as follows:

$$\begin{aligned} \beta &= \int_0^\infty 5.u_v.x.e^{\frac{5x}{4}}.e^{e^{-u_v.x}-1}.\frac{e^{-u_v.x}}{4}dx, \\ b^{(2)} &= \int_0^\infty 2x^2 e^{e^{-2x}-1}.e^{2x}dx, \\ S_v^*(\theta) &= \int_0^\infty u_v.e^{e^{-u_v.x}-1}.e^{\theta x}.e^{-u_v.x}dx. \end{aligned} \tag{10}$$

The values given in [8] account for the rest of the variables in equations (5) and (6).

IV. The Proposed Genetic Algorithm

Below follows a brief discussion of the theoretical background to GA, together with the design and implementation of the GA for the queuing model.

A. Theoretical Background

The individuals in the initial population are a solution to the problem that requires resolution and constitute the parameters for the model. The fitness function in the GA is responsible for determining how good an individual is as a solution to the problem [19]. This is a vital step in the GA as the fitness score given to the individuals in the population will determine which individuals will be selected for reproduction. The process of choosing the best individuals (with the best fitness scores) is known as "natural selection" and selects two of the "strong" individuals from the population as parents [20].

Once the selection process has occurred, the crossover phase in the GA can be processed. This is where attributes from the two parents are combined into a single new individual for the next generation. This new individual is known as the offspring of the parents. When the offspring is formed, some parts of their genes can be mutated (changed). This is usually a low probability and does not significantly change the entire individual. The mutation is done to prevent early convergence of the algorithm (that is, it maintains the diversity within the populations) [21]. The process is then repeated for the newly formed generation and then repeated until some stopping condition or if the population has converged. This means that the individuals in a generation do not produce offspring that are different from the parents [22].

B. Design of the GA for the $M/(M_1, M_2)/1$ Model

The population in a GA starts with a randomized set of solutions referred to as chromosomes or individuals. These individuals in the population then undergo crossovers and mutations, which are then used to populate the generation further. Using the fitness function a fitness value was obtained for each individual of the population [19]. The individuals with the highest fitness values are then used to populate the next generation by choosing the best parents to produce offspring. This process is known as the selection process and parallels the process of natural selection. After multiple iterations of generations, the best-performing individuals/chromosomes in the populations converge to an optimum or sub-optimum solution [22], [23].

1) **Assumptions:** It is assumed that the system that is being optimized uses a M/(M1,M2)/1 queue. It is also assumed that the queue has a Poisson arrival rate of λ. Assuming that the server is in a vacation period, the server will continue to operate at a significantly lower rate [24]. It is also assumed that at service completion if there is a queue, the server will go back to normal working rates. The vacation time is assumed to be exponentially distributed with a rate of θ [8].
A largely significant assumption is that the service times, inter-arrival times, and working vacation times are mutually independent of one another and that the service order is first-in-first-out [8].

2) **Input parameters and decision variables:** The model requires the input parameters for the population size, pairs of chromosomes that will be crossed over, the number of chromosomes that will be mutated, and the upper and lower boundaries for the ρ and θ values.

3) **Crossover and mutation operators:** The population size can be stated as 'p'. For this implementation, a population size of 100 was chosen as it offers enough variety in the population for the GA to work with. The solution precision is chosen to be 5 decimal places with upper and lower bounds of -9 and 9, therefore each member in the population is a 40-bit binary number because we have two variables θ and ρ. The population is generated with 40 random bits per chromosome. An example of this is shown in Fig. 2:

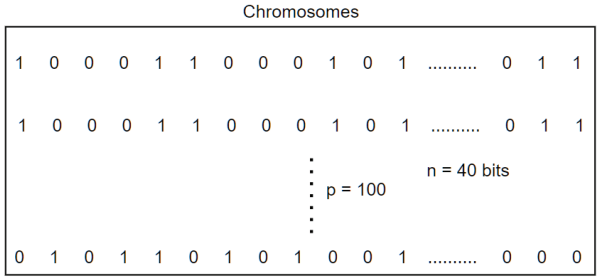


Fig. 2: Population containing 100 chromosomes of 40 bits

Once the first generation has been populated the crossover process can commence. In the crossover process, a pair of chromosomes is randomly picked from the population. The pair of chromosomes are called 'parents' and these parents are split at a random point between the first and 40th bit. All the elements after this random point are swapped between the two parents. The two outputs are then offspring 1 and 2. These offspring are then added to the current population. An illustration of the crossover is shown in Fig. 3:

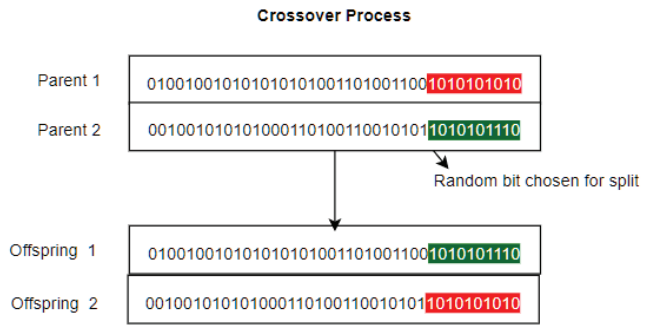


Fig. 3: An illustration of the crossover process.

During the mutation process, n amount of chromosomes are picked and a random bit within that chromosome is flipped. In the implementation for this GA, $n = 30$, and therefore 30 mutations were done on the population. The mutation process is illustrated in Fig. 4:

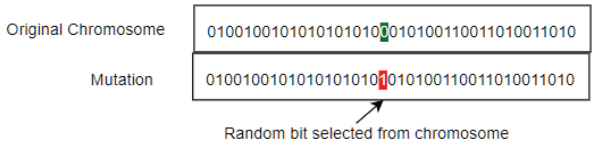


Fig. 4: Illustration of the mutation process.

4) **Chromosome evaluation:** Once the crossover and mutation processes are done, they are added to the current population. This entire population can then be evaluated and given fitness values. To achieve this, the 40-bit binary numbers that represent each chromosome in the population are converted to a ρ and θ value between 0.01 and 3. This is done by splitting the 40-bit binary number into two 20-bit binary numbers. The two binary numbers are then converted into decimal numbers and the following equations are used for converting the decimal number into the correct ρ and θ values,

$$\rho = \text{lb} + A \times \left(\frac{\text{upper bound} - \text{lower bound}}{2^{20} - 1} \right), \tag{11}$$

$$\theta = \text{lb} + B \times \left(\frac{\text{upper bound} - \text{lower bound}}{2^{20} - 1} \right). \tag{12}$$

In the equations above 'lb' is defined as the lower bound, A and B are the ρ and θ values are converted to decimal representation. The ρ and θ values are then substituted into equations 9, 10, 11, and finally equation 1. The μ_v value chosen for this implementation was 1, but could be chosen as any value. 1 was chosen as it was found to be the average vacation rate that is used in [8]. The output of the M/G/1 working vacations model is then used as the measure of fitness. A higher fitness equates to a lower queue length as the algorithm optimizes for the lowest queue length. Each chromosome in the population is then given a fitness value and these fitness values are used in the selection process. In the selection process, e number of the best chromosomes are selected from the current population for the next generation. These chromosomes have the highest fitness values associated with them.

The remaining population is then used to determine the selection probability which divides the fitness values by the sum of all fitness values in the population. The cumulative probabilities can then be found for the population. A random value is then generated and compared to the cumulative probability value of a chromosome to confirm or reject that chromosome in the new generation. The best-performing chromosome in the generation is then used to determine the ρ and θ values that should be used in the M/(M1,M2)/1 working vacation and vacation interruption queuing model. The entire process is then repeated for the next generation until the algorithm converges to a solution.

5) **Stopping condition:** The stopping condition for the GA is when the change of queue length is smaller than 5 percent for 10 iterations.

6) **Flowchart of the GA:** The entire procedure of the GA as explained above is illustrated in Fig. 5 below. The initialization begins with an initial population which is randomly generated where each individual in the population is

represented as a 40-bit binary number, encoding the parameters ρ and θ. The population size is set, the number of generations, the crossover pairs and the number of mutations. The population is then generated after which the algorithm enters an iterative loop. The crossover operation is performed on the crossover pairs, where a random crossover point is chosen and the bits are swapped between the two parents to create the offspring. Mutation is then applied to the individuals by randomly flipping a bit in their binary representation. Each individual in the population is evaluated for its fitness using the $M/(M_1, M_2)/1$ model where the goal is to minimize the queue length. This is possible because each 40-bit chromosome is converted into its corresponding ρ and θ values. The chromosomes with the highest fitness values are then directly selected for the next generation. The remaining chromosomes are selected based on their fitness probabilities. This method ensures that better-performing chromosomes have a higher chance of being selected, but it also allows for diversity by giving less-fit chromosomes a chance to be selected. The process of selection, crossover, mutation, and evaluation is repeated for a specified number of generations.

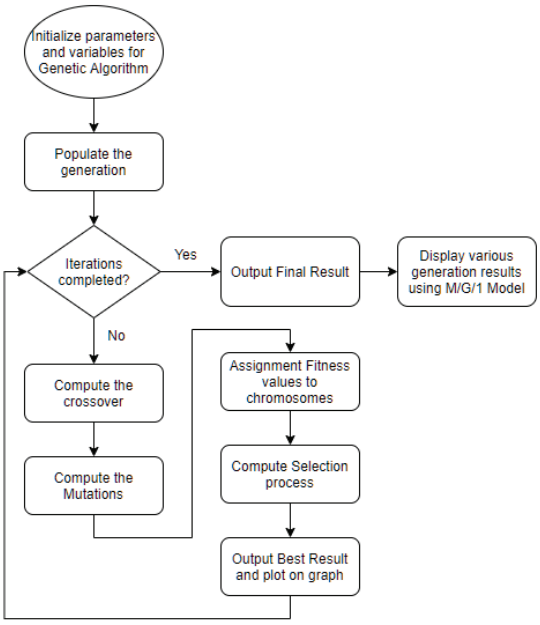


Fig. 5: Genetic Algorithm flow diagram.

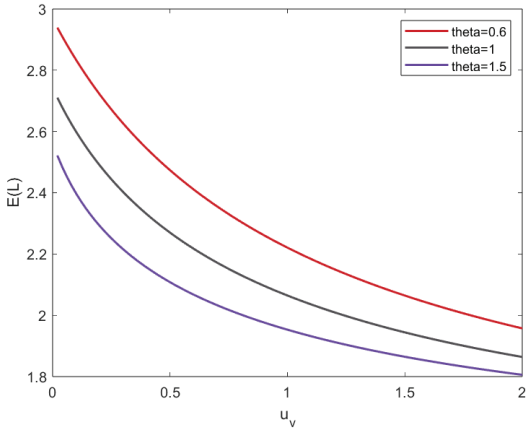
V. Simulation Results

A. Simulation Setup

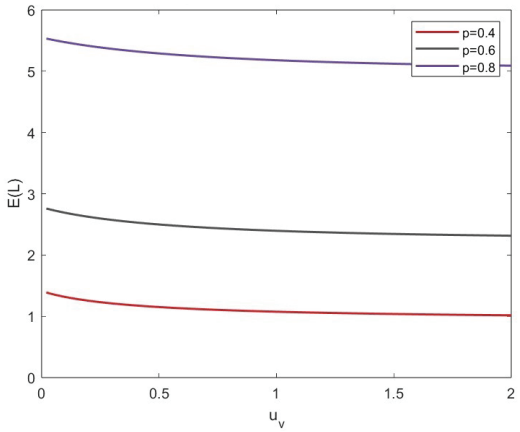
In order to replicate the numerical results from [8], an exponential distribution was assumed for both the service time during a busy period and the service time during a working vacation period. During the busy period $\mu_b = 2$ and for the vacation period μ_v was a value between 0 and 2. The mean of the busy period distribution was $\frac{1}{\mu_b}$ and for the vacation period was $\frac{1}{\mu_v}$. The values for θ and ρ were then adjusted while the other parameters of λ, μ_b and either θ or ρ were fixed.

B. Base Queuing Model Results

In order to replicate the expected queue length against u_v in $M/(M_1, M_2)/1$ queuing model for varying θ values, the following is given with $\theta = 0.6$ or 1 or 1.5, $\lambda = 1.25$, $\rho = \frac{\lambda}{\mu_b}$, where $\mu_v \in (0, 2)$. In order to replicate the expected queue length against u_v in $M/(M_1, M_2)/1$ model for varying ρ values and state probability values, the ρ and θ values are given. To obtain the values for β, $b^{(2)}$ and $S_v * (\theta)$, numerical integration using variable precision was used which directly approximated the mean. This simplifies the complexity of the mathematics and reduces computation time, however, discards the variance information. The depiction of the expected queue length with respect to varying ρ and θ values is shown in Fig. 6 below:



(a) Expected queue length for varying θ

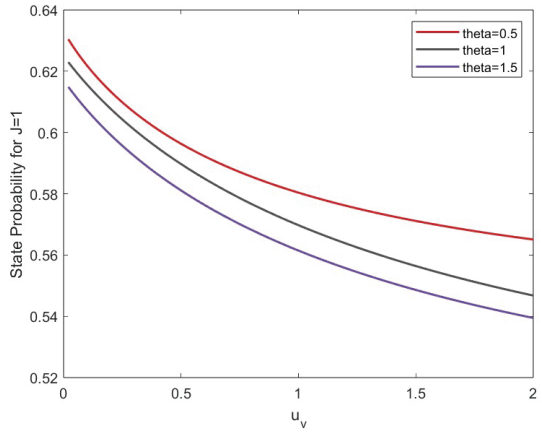


(b) Expected queue length for varying ρ values

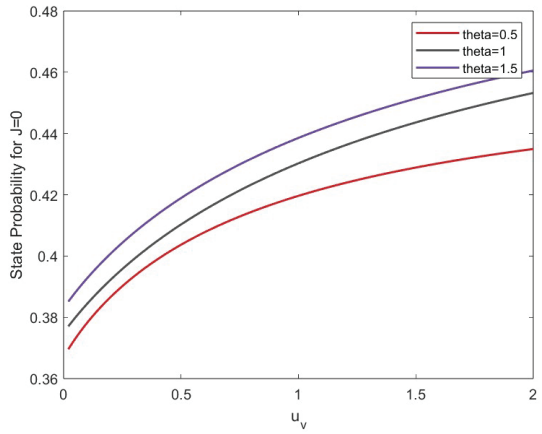
Fig. 6: The state probability of the server against u_v in M/(M_1, M_2)/1 model for varying θ and ρ values replicated in Matlab.

As seen in Fig. 6, it is clear that the expected queue length decreases as the vacation service rate increases. This is expected as a higher vacation service rate would eventually converge to the busy service rate. It is also seen that as theta increases, the expected queue length decreases. This effect of theta on the expected queue length diminishes as the vacation service rate increases. It can be seen in Fig. 6 that at $\mu_v = 2$ the differences between the rate of service (θ) become negligible and has a

much higher effect at lower values of μ_v. When considering the bottom graph in Fig. 6, it is clear that increasing the value of traffic intensity, ρ, increases the expected queue length. In practical applications, this is to be expected as the rate of arrival will increase. It is also seen that the effect of ρ does not seem to be changed with a variation of μ_v. The distances between the three curves in Fig. 6 remain relatively constant throughout an increasing value of μ_v. The state probabilities for normal service and vacations are shown in Fig. 7:



(a) State probability for varying θ at J = 1



(b) State probability for varying θ at J = 0

Fig. 7: The state probability of the server in normal service and vacation period against u_v in M/(M_1, M_2)/1 model for varying θ values.

It is clear from Fig. 7 above that for the normal service time (J = 1), decreasing θ results in lower expected queue lengths while the opposite happens for the vacation service time (J = 0). This is because as the vacation service rate increases, the probability of the system staying in a vacation state is higher as the expected length of the queue is less likely to increase. The opposite is true for the busy server because as the vacation service rate increases, the system is less likely to be in a busy period as the expected queue length decreases.

C. Implementation Using the Proposed GA Model

The fitness values (scaled to ensure no negative values for computation) are plotted against each generation are shown in Fig. 8:

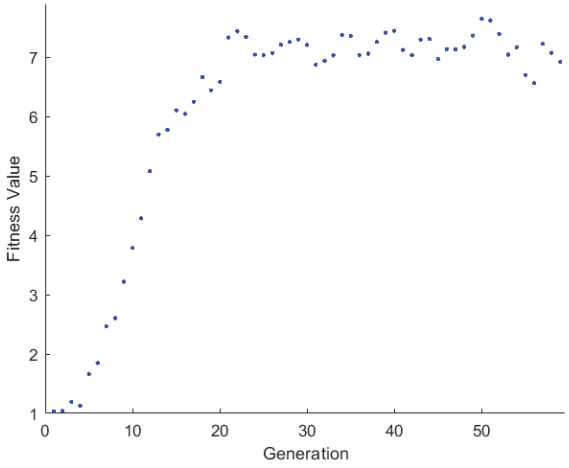


Fig. 8: Fitness values for each generation.

After each selection process is computed for every generation, the best-performing chromosomes are printed to the command window to show the ρ and θ values that performed the best. A higher fitness value is associated with a lower expected length of the queue for the given ρ and θ values. It is clear from Fig. 8 that the algorithm is converging to a solution and no more than 20 generations are needed. The expected queue length against the vacation rates for different generations is shown in Fig. 9:

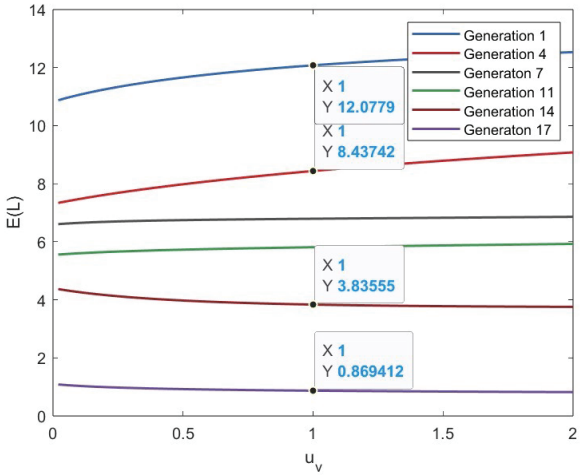


Fig. 9: Expected queue length against u_v for $M/(M1, M2)/1$ for different generations

In Fig. 9 above, it is clear that with each generation there is an improvement in the expected queue length as it decreases. After 20 iterations, the expected queue length converges for the given boundaries for ρ and θ. According to the GA output, the best-performing values were $\rho = 0.4792$ and $\theta = 2.2107$.

D. Comparison of Base Model and Proposed Model

The GA was successfully implemented, as shown in Fig. 10:

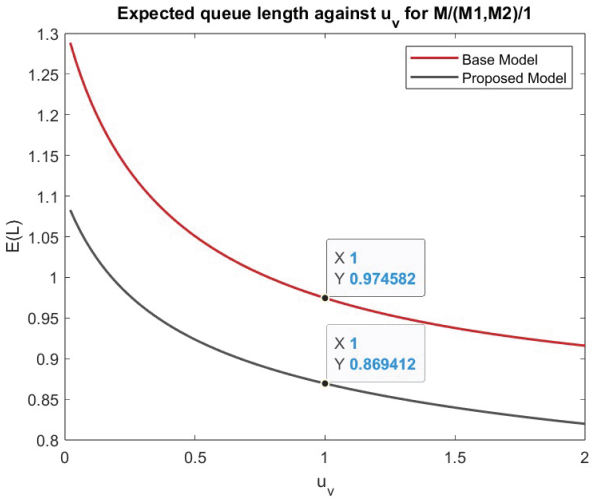


Fig. 10: Comparison of Base Model and Proposed Model

From the results, it is clear that the output from the GA has a lower expected length than the best-performing result from [8] for all vacation rates between 0 and 2. The results from the proposed GA model offered an average of 10.79% lower expected length for the queue. As can be seen in Fig. 10, the results produced were superior to the queue length results in [8]. The lowest length of queue from [8] at $\mu_v = 1$ was 0.97 and the optimum result from the output of the GA was a queue length of 0.869 for $\mu_v = 1$. The main advantage of using the genetic algorithm is that the algorithm will optimize the queue length of any given set of parameters. In this case, the algorithm was designed to optimize for θ and ρ values at some value of μ_v. This algorithm can easily be adapted to any values of μ_v with a boundary for the ρ and θ values. The GA will also be able to optimize for the queue length if μ_b is unknown within a range. This allows for flexibility in the design of systems searching for specific lengths of queues.

VI. Conclusion

In this article, the design and implementation of an $M/G/1$ queue with working vacations was proposed using GA. The overarching motivation for the improvement to the model was to provide flexibility in the design, as well as lower expected queue lengths for $M/(M1, M2)/1$ queuing systems. The implemented GA successfully optimized the M/(M1,M2)/1 queuing model with working vacations and vacation interruptions, and achieved optimal results for the given set of parameters. Compared to the representative baseline [8], our proposed algorithm showed superior performance with a lower queue length. Specifically, the GA algorithm achieved a 10.79% reduction in average queue length compared to the baseline. Overall, this study demonstrates that the proposed GA algorithm can effectively optimize the design of queuing systems and significantly reduce queuing delays in various applications, including wireless communication systems.

The GA allows the M/(M1,M2)/1 analysis model to be used for designing systems and providing greater robustness and design adaptability, allowing for more precise control of the system [23]. By providing a possible range of parameters for the queuing model, the values for optimum design will be computed using the GA and the model that was developed in [8].

A. The Advantages of GA in Wireless Communications

In wireless communication systems, particularly in cognitive radio networks where efficient channel utilization is crucial, the proposed GA can be applied to solve queuing problems and optimize the design of cognitive radio systems. By incorporating the GA, cognitive radio systems can achieve better resource utilization and reduced queuing delays, leading to improved overall performance. Specifically, in cognitive radio networks, the GA can be used to enhance spectrum allocation and dynamic channel assignment, thereby enabling the system to adaptively allocate channels based on the current traffic conditions, optimizing the utilization of available resources. By minimizing queuing delays and maximizing throughput, the GA improves the quality of service and enhances the overall efficiency of cognitive radio systems.

B. Future Research Work

The trade-off with using the GA is the extremely high computation cost. Further investigation will have to be done to optimize for the population size, number of generations, number of mutations and crossover chromosomes, and the number of elite chromosomes that are put into the next generation. All of these factors can increase or decrease the computational time and accuracy of the GA. The GA can be further extended to include different probability distribution functions for the queuing model in the population. This would allow for the optimal output between using different distributions for the busy service rates.

References

[1] S. R. Chakravarthy, Shruti, and R. Kulshrestha, "A queueing model with server breakdowns, repairs, vacations, and backup server," *Operations Research Perspectives*, vol. 7, p. 100131, 2020.

[2] M. Choudhary and P. Kaur, "Application of queuing theory in computer networks: A review," *International Journal of Advanced Science and Technology*, vol. 30, no. 4, pp. 274–282, 2021.

[3] A. Srinivasan and P. Karthikeyan, "Queuing theory application in traffic management at a four-way signalized intersection," *International Journal of Advanced Research in Engineering and Technology*, vol. 11, no. 5, pp. 455–460, 2020.

[4] R. Shah and M. Zaki, "Application of queuing theory in healthcare: A systematic review," *International Journal of Healthcare Management*, vol. 13, no. 1, pp. 47–58, 2020.

[5] S. Bhatnagar, S. Rao, and S. Sharma, "Queuing theory: A systematic review," *International Journal of Recent Technology and Engineering*, vol. 8, no. 6, pp. 2321–2328, 2020.

[6] F. Palunčić, A. S. Alfa, B. T. Maharaj, and H. M. Tsimba, "Queueing models for cognitive radio networks: A survey," *IEEE Access*, vol. 6, pp. 50 801–50 823, 2018.

[7] M. Jain and A. Jain, "Working vacations queueing model with multiple types of server breakdowns," *Applied Mathematical Modelling*, vol. 34, no. 1, pp. 1–13, 2010.

[8] M. Zhang and Z. Hou, "Performance analysis of m/g/1 queue with working vacations and vacation interruption," *Journal of Computational and Applied Mathematics*, vol. 234, no. 10, pp. 2977–2985, 2010.

[9] P. K. Agrawal, A. Jain, and M. Jain, "M/M/1 Queueing Model with Working Vacation and Two Type of Server Breakdown," *Journal of Physics: Conference Series*, vol. 1849, no. 1, p. 012021, Mar 2021.
[10] S. Majid and P. Manoharan, "Analysis of an M/M/1 Queue with Working Vacation and Vacation Interruption," *Applications and Applied Mathematics: An International Journal (AAM)*, vol. 14, no. 1, p. 2, 2019.
[11] R. Manikandan and S. S. Nair, "An M/M/1 Queueing-inventory System with Working Vacations, Vacation Interruptions and Lost Sales," *Automation and Remote Control*, vol. 81, no. 4, pp. 746–759, Apr 2020.
[12] R. Tian, L. Hu, and X. Wu, "Equilibrium and Optimal Strategies in M/M/1 Queues with Working Vacations and Vacation Interruptions," *Mathematical Problems in Engineering*, vol. 2016, pp. 1–16, Jan 2016.
[13] P. K. Joshi, S. Gupta, and K. N. Rajeshwari, "A Study of Steady State Analysis of D/M/1 Model and M/G/1 Model with Multiple Vacation Queuing Systems," *South East Asian J. of Mathematics and Mathematical Sciences*, vol. 16, no. 1, pp. 37–50, 2020.
[14] E. Kasim and G. Gupur, "Functional Analysis Method for the M/G/1 Queueing Model with Single Working Vacation," *Open Mathematics*, vol. 16, no. 1, pp. 767–791, Jan 2018.
[15] R. Kalyanaraman and A. Sundaramoorthy, "A Multi Server Markovian Working Vacation queue with Breakdown, N-policy and with Server State Dependent Rates," *Journal of Mathematics*, vol. 9, no. 2, pp. 21–28, Mar 2022.
[16] R. Bhat and A. Shukla, "A study of m/g/1 and m/ma/1 queuing models," *International Journal of Computer Applications*, vol. 179, no. 20, pp. 17–20, 2020.
[17] B. Chen, *Queueing Theory: Modeling and Applications in Science and Engineering*. Springer, 2019.
[18] L. Tao, L. Zhang, X. Xu, and S. Gao, "The GI/Geo/1 queue with Bernoulli-schedule-controlled vacation and vacation interruption," *Computers & Operations Research*, vol. 40, no. 7, pp. 1680–1692, 2013.
[19] D. E. Goldberg, *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley Longman Publishing Co., Inc., 1989.
[20] M. Mitchell, *An introduction to genetic algorithms*. MIT Press, 1998.
[21] J. H. Holland, *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. University of Michigan Press, 1975.
[22] Q. Gao, K. Qi, Y. Lei, and Z. He, "An improved genetic algorithm and its application in artificial neural network training," in *2005 5th International Conference on Information Communications Signal Processing*, 2005, pp. 357–360.
[23] P. Guo, X. Wang, and Y. Han, "The enhanced genetic algorithms for the optimization design," in *2010 3rd International Conference on Biomedical Engineering and Informatics*, vol. 7, 2010, pp. 2990–2994.
[24] P. Rajadurai, M. C. Saravanarajan, and V. M. Chandrasekaran, "Analysis of an M/G/1 feedback retrial queue with unreliable server, non-persistent customers, single working vacation and vacation interruption," *International Journal of Services and Operations Management*, vol. 24, no. 2, pp. 235–266, 2016.



Mduduzi C. Hlophe (Member, IEEE) received the M.Sc. degree in electrical and electronic engineering from the University of Johannesburg, Johannesburg, South Africa, in 2015, and the Ph.D. degree in electronic engineering from the University of Pretoria, Pretoria, South Africa, in 2020. He is currently a Postdoctoral Research Fellow with the Broadband Wireless Multimedia Communications (BWMC) group, Department of Electrical, Electronic and Computer Engineering, University of Pretoria. His research interests include machine learning and applied mathematical modeling for wireless communications—spanning cognitive radio networks, spectrum management, and resource allocation—as well as predictive analytics and inference with applications in finance, health, and robotics.



Daniel Naicker (Student Member, IEEE) received his BEng and BEng (Hons) degree in Electronic Engineering from the University of Pretoria, South Africa. He is currently completing his M.Sc. in engineering at the same institution. His research focuses on deep learning for cognitive radio networks and signal classification.



Bodhaswar T.J. (Sunil) Maharaj (Senior Member, IEEE) received the Ph.D. degree in engineering in the areas of wireless communications from the University of Pretoria. He is currently a Full Professor and holds a research position in Broadband Wireless Multimedia Communications, Department of Electrical, Electronic and Computer Engineering, University of Pretoria. His research interests include OFDM MIMO, cognitive radio resource allocation, edge computing and 5G/6G cognitive radio sensor networks.

CONTENTS PAGE